COPS 프로토콜을 이용한 정책 기반 네트워크 연구

김영하*, 육동철**, 박승섭**
*부경대학교 전산정보학과, **부경대학교 전자계산학과
e-mail : ipakp@hotmail.com

A Study On Policy Based Network Using COPS Protocol

Young-Ha Kim*, Dong-Cheol Yuk**, Seung-Seob Park**
*Dept. of Computer Science Industry, Pukyoung National University
**Dept. of Computer Science, Pukyong National University

요약

인터넷의 폭발적인 성장과 함께 미래의 네트워크에서 필수적으로 지원되어야 할 것이 QoS이다. 그러나 인터넷 환경은 현재 새롭게 등장하는 실시간, 멀티미디어, 그리고 멀티캐스팅 서비스를 제공하기 위한 응용 프로그램들을 지원하기에는 적당하지 않다. 또한 양질의 서비스를 받고자 하는 사용자들의 요구 역시 만족시키기 어렵다. 그러므로, 원하는 서비스의 질을 제공하기 위해서는 무엇보다도 네트워크 기반 구조가 이를 제공하여야 한다.

본 논문은 Linux 기반 환경에서의 정책을 적용할 수 있는 COPS의 구현과 정책의 서버에 해당하는 PDP와 클라이언트인 PEP 구현, 서버와 사용자간의 인터페이스 구현에 초점을 두었다.

1. 서론

COPS(Common Open Policy Service) 프로토콜은 네트워크 정책 서버와 클라이언트 사이의 정책 정보를 교환하기 위해 사용되는 질의응답 프로토콜이다. COPS는 정책을 반영하기 위한 QoS(Quality of Service) 도구의 일환으로 사용된다.

우선, QoS는 네트워크망에서 다른 패킷들과 비교하는 방법으로 패킷의 흐름 또는 어떤 종류의 패킷을 다루기 위한 목적으로 패킷들을 분류화 하는 것과 관련성이 있다[1]. 그래서, QoS를 가능하게 하기 위해서는 IntServ(Integrated Service)[2]를 위한 RSVP(Resource reServation Protocol)와 DiffServ [4] 서비스와 COPS등 여러 프로토콜 사용을 요구한다.

본 논문에서는 서론에 이어서 2장 관련연구에서는 정책기반네트워크의 구조와 기능, COPS 프로토콜의 형식과 메시지 동작, LDAP 디렉토리 서버에 대해서 서술한다. 3장에서는 구현 방안으로 작업 망 모델, PEP구성, PDP 구성 결과에 대해 서술한다. 마지막으로 4장에서는 결론 및 향후과제를 제시한다.

2. 관련연구

본 장에서는 정책 기반 네트워크에서의 개요와 정책의 구조와 기능, COPS 프로토콜의 기본 모델, 프로토콜 형식, 통신방법에 대해서 설명한다.

2.1 정책 기반 네트워크

인터넷의 폭발적인 성장과 함께 미래의 네트워크에서 필수적으로 지원되어야 할 것이 바로 QoS이다. 기존의 IP 기반의 인터넷 기반 구조는 서비스의 질을 보장하기에는 어려움이 따른다. QoS를 만족시킬 수 있는 서비스 모델의 정의가 필요하고, 각 서비스의 모델들을 지원하기 위한 기반으로써 우선 기존의 라우터에 새로운 기능 추가 및 이를 뒷받침해 줄 수 있는 프로토콜들이 구현되어야 한다.

2.1.1 정책 기반 네트워크의 개요

네트워크 망 관리를 우수한 메커니즘의 기반으로 네트워크 관리자가 직접 관여하여 정책을 적용해야 할 필요가 있다. 좋은 서비스와 서비스 레벨에 따른 인증이 필요하다. 이를 정책으로 규정하여 반영 할 수가 있다. 정책은 단순한 의미에서 특정 조건이 있을 때, 동작으로 발생되는 하나 이상의 규칙이다[5].

정책은 때때로 다른 여러 규칙으로 구성될 수 있다. 그래서 사실상 정책은 정책을 포함한다. 복잡한 정책들이 단순한 정책처럼 구현될 수 있으며, 관리를 단순화 시킬 수 있는 계층적 개념으로 표현될 수 있다[1].

2.1.2 정책 구조와 기능

IETF의 RAP(Resource Allocation Protocol) 작업그룹은 RSVP(Resource Reservation Protocol)와 이를 가능케한 IntServ(Integrated Service)에 안정된

정책 제어 모델 설립을 추진하였다[2][15][16].

정책 기반 프레임워크은 크게 두 개의 구성으로 정책서버인 PEP(Policy EnforcementPoint)와 클라이언트로 PDP(Policy Decision Point)로 나뉘어져 있다. PEP는 직접적인 동작을 집행하고, PDP는 법의 근거를 바탕으로 결정을 내린다는 의미이다. PDP는 정책 저장소, 인증 서버 또는 다른 엔티티에서 정책에 대한 정보를 갱신 및 수정을 할 수 있다.

그림1은 PEP, PDP, 정책 저장소 등의 세분화는 물리적 분리가 아닌 논리적 세분화 기능을 기반으로 하고 있다. PEP는 PDP의 수행이 불가능할 때, 그 이전의 백업용으로 내부적으로 정책 결정 수행하는 LPDP(Local PDP)가 포함 될 수도 있다. 그리고 보안상 PEP는 PDP의 마지막 결정을 참조하는데, 이 요구를 위해 request 메시지를 지속적으로 전송한다.

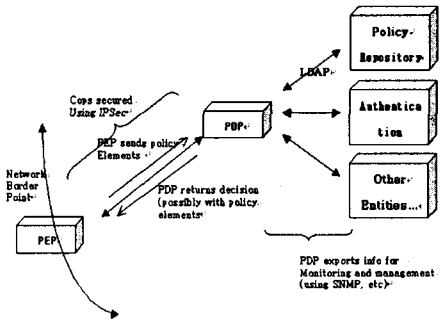

그림 1. 정책 기반 프레임워크

2.2 COPS프로토콜

본장에서는 COPS 프로토콜의 기본 모델과 프로토콜 형식, PEP와 PDP간의 통신 방법에 대해서 서술한다.

2.2.1 COPS의 기본 모델

COPS는 PEP와 PDP 사이의 정책 정보를 교환하기 위해 사용하며, 임의의 LPDP는 PDP가 없을 때 local policy의 결정들을 내리기 위한 장치로 사용된다.

COPS는 PEP와 PDP사이의 정책 정보를 교환하기 위해 사용된다. PEP는 지속적인 TCP 연결로 request 메시지를 보내고, PDP로부터 decision 메시지를 수신한다. request 메시지가 PDP에 도착되어 decision 메시지를 수신하면 configuration data를 PEP에 성공적으로 설치를 할 수 있고, 설치 승인은 report 메시지를 통해 PDP에 전달된다. 오류 검출은 KA(keep-alive)메시지를 통해 연결되어 있다는 것을 PEP와 PDP가 계속 확인 할 수 있다. 오류가 검출될 때는, PEP와 PDP에 연결을 다시 설정하거나, 다른 PDP와 연결을 다시 시도한다.

2.2.2 메시지 내용

PEP와 원격 PDP의 메시지뿐만 아니라 PEP와 원격 PDP 사이의 기본적인 메시지 교환을 설명한다. 참고사항으로 무결성 객체가 포함되어 있다면, 이는 메시지의 마지막 객체이다. 메시지가 보안이 요구되는 상태에서 무결성 객체를 가지지 않은 메시지를 수신하게 되면 메시지를 수신한 쪽은 고유한 에러코드가 명시된 client-type=0인 client-close 메시지를 전달해야 한다.

- Request(REQ) PEP→PDP

stateful 핸들이 새로운 요구를 설정하게 되면, 요구에 대한 수정은 이전에 설치된 핸들을 지정하는 REQ 메시지를 사용하여 만들 수 있다. PDP는 핸들과 교환된 정보를 조사하고 주어진 클라이언트타입에 대한 연결을 위하여 Decision을 사용한다.

- Decision(DEC) PDP→PEP

PDP는 REQ의 응답으로 DEC메시지를 전달한다. 같은 클라이언트 핸들과 Context 객체와 Decision 플래그 타입 객체와 관련된 여러개의 decision 객체를 PEP에 전송한다.

- Report State(RPT) PEP→PDP

PDP의 decision를 수행하는 성공과 실패를 PEP가 PDP에 전달 사용되어진다. 상태변화와 관련된 계산치를 보고한다. 임의의 ClientSI(Client Specific Information)와 report의 형태로 지정되어진 Report-Type를 Client-Type 하나 당 추가적으로 정보를 제공한다.

- Client-Open(OPN) PEP→PDP

PEP에 의해서 PDP에 연결 요청시 사용되어진다. PEP에 지원되어지는 클라이언트 타입들에 의해서도 사용되어진다.

- Client-Close(CC) PEP→PDP, PDP→PEP

특별한 클라이언트의 타입이 더 이상 지원되지 않는다는 것을 PDP 혹은 PEP에 통보하기 위해 생성시킬 수 있다.

- Keep-Alive(KA) PEP→PDP, PDP→PEP

연결을 위해 수신된 모든 CAT메시지에서 명시된 모든 KA타이머의 최소 값으로 정의된 주기 내의 PEP에 의하여 전송되어져야 한다.

- Synchronize sate Complete(SSC) PEP→PDP

PDP가 동기화 단계의 요구를 PEP로 보낸 후 PEP가 PDP로 보낸다. PEP는 동기화를 끝낸다. 이것은 모든 클라이언트의 상태가 성공적으로 다시 요구되었을 때 PDP가 알 수 있도록 하기 위해 유용한 것이며, PEP와 PDP는 동기화가 완전하게 연결된다.

3. 정책 기반 네트워크의 구현

COPS 프로토콜의 구현을 위해서 Vovida에서 제작한 COPS 프로토콜 객체를 사용하였다. 그리고 정책 서버와 정책 저장소간의 통신에 사용되는 LDAP 프로토콜은 openLDAP를 사용하였다. 자원 예약을 위한 RSVP 데몬을 대신할 임의적 서비스 시스템과

PEP 클라이언트, 정책서버와 연결되는 LDAP 서버와 정책 스키마 구현을 위한 작업은 여러대의 퍼스널 컴퓨터에 와우리눅스(버전 7.0과 7.1)를 사용해서 작업하였다.

3.1 Service System의 구성

서비스 시스템구성은 필요한 값만 준다고 가정할 때, 값만 주고 받을 수 있는 프로세스를 생성하여, 콘솔상에서 값을 입력하는 방식으로 구현하였다.

그림8은 메시지 신호 처리 흐름도를 나타내고 있으며, 메시지 프로세싱 방식을 사용해서 PepAgent와 시스템 서비스간의 값을 공유해서 두 프로세스간 통신을 할 수 있다.

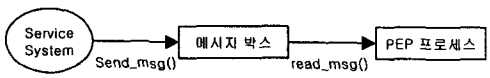

그림2. 메시지 신호 처리 흐름도

메시지 큐를 다루기 위해 클래스 MSG Class를 정의하였고, send_msg, read_msg 메서드는 qid를 식별해서 사용하였다. 메시지들은 차례대로 큐로 보내지고 큐에서 여러 가지 다른 방법으로 메시지들이 조회된다. 각각의 메시지큐는 IPC 확인자에 의해 유일하게 확인되어 진다. 메시지 큐를 다루기 위해 클래스 MSG Class를 정의하였고, send_msg, read_msg 메서드는 qid를 식별해서 사용하였다.

그림 3는 두 개의 메시지가 메시지 큐에 저장된 결과를 나타낸 것이며, 그림 4는 두 개의 메시지가 메시지 큐에 저장된 결과를 나타내고 있다. PepAgent는 read_msg()메서드를 사용해서 메시지 큐를 검사하여 메시지가 있을 경우 정책 결정을 위해 PdpAgen와 동작을 수행한다.

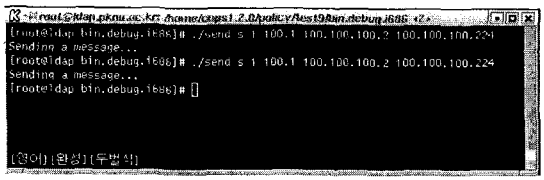

그림 3. 두 개의 메시지 전송

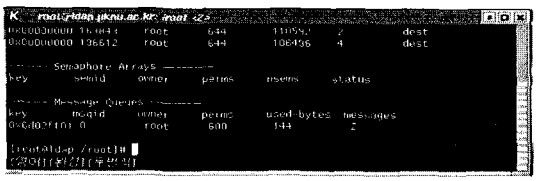

그림 4. 메시지 큐에서의 두 개의 메시지 확인

3.2 PEP클라이언트 구성

네트워크 노드는 두 가지 기능을 하는 컴포넌트로 구성되어 있다. PEP는 항상 네트워크 노드 내부에 위치해야 하는 반면에 PDP는 어느 곳에서든지 위치할 수 있다. 그림 5은 PEP는 네트웍 노드의 컴포넌트이며, PDP는 분리된 정책서버 안에 있는 엔티티이다. 한 개의 시스템에서 서버와 클라이언트간 통신이 가능하다.

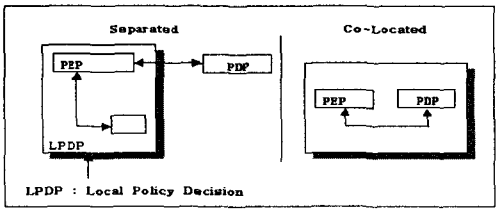

그림 5. Enforcement and Decision Point Scenarios

각 메시지의 통신 중에 KA메시지는 클라이언트와 서버간 연결이 설정되면 지속적으로 통신을 하며, KA 메시지 통신을 위해서 스레드 방식으로 처리하였다.

그림 6은 PepAgent와 PdpAgent간 client-open 메시지와 client-open 메시지 연결 설정 처리한 후 Keep Alive 메시지 설정까지의 동작 결과를 나타낸 결과이다.

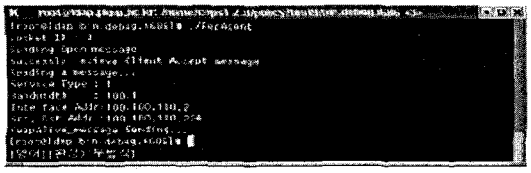

그림 6. 메시지 박스에 대한 PepAgent의 Client Accept 및 KA 메시지 수신 동작 결과

3.3 PDP 서버 구성

PDP는 정책 결정을 위해 PEP와 COPS 프로토콜을 사용해서 통신한다. PEP와 통신할 때는 소켓 프로그래밍으로 통신하며 PDP 프로세스가 활성화되면, PEP에서 연결 요청이 올때까지 listen 상태에 들어간다. 이때 PDP 프로세스는 여러개의 PEP로부터 소켓 연결 요청을 받는다.

그림 7은 일대 의 통신을 처리하기 위해 소켓 번호에 따른 서버와 클라이언트 연결은 vector 함수와 map 라이브러리 함수로 관리한다.

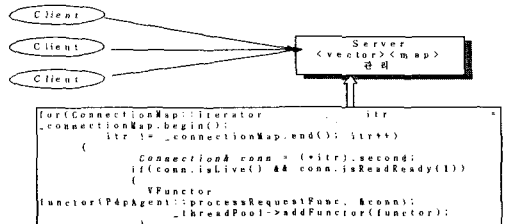

그림 7. 여러개의 PepAgent와 PdpAgent의 관계

그림 8는 각 클라이언트의 연결 소켓 번호에 따른 처리를 흐름도와 서버에서 <vector>와 <map> 라이브러리로 처리하는 과정을 나타내었다.

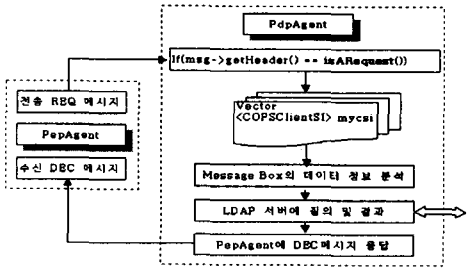

그림 8. REQ 메시지와 DEC 메시지의 처리 흐름도

그림 9은 PepAgent에서 REQ 메시지가 왔을때의 동작에 대한 흐름을 나타내고 있는 그림이다. PepAgent는 PdpAgent에 정책을 요청하고 PdpAgent는 LDAP 서버에 질의해서 그 결과를 PepAgent로 전달한다. 그러나 PepAgent와 PdpAgent간의 메시지 통신에서 직접적으로 LDAP 서버에 접근할 수 있는 계기가 되는 COPS 메시지 프로토콜의 REQ 메시지이다. PdpAgent 프로세스는 MSG 클래스의 메시지 박스의 내용이 REQ 메시지의 SI정보에 포함하여 전송하며, 이 정보를 토대로 질의를 형성한 후에 LDAP 서버에 질의한다.

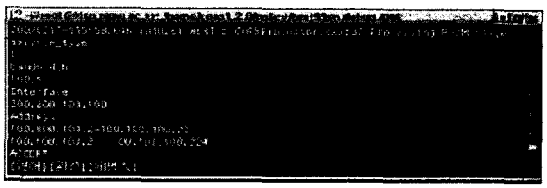

그림 9. PdpAgent가 REQ 메시지 처리에 대한 결과

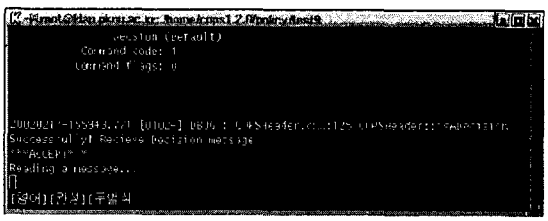

그림 10. DEC 메시지에 대한 결과와 그 다음 서비스 메시지 대기 상태의 PepAgent

그림 10은 PepAgent가 PdpAgent에 REQ메시지를 전송하였을 때, PdpAgent가 메시지 처리에 대한 결과 그림이다. 그림 18은 PepAgent가 DEC 메시지를 수신하고, 정책에 대한 결과가 Accept임을 나타낼 결과 그림이다.

4. 결론

현재 인터넷 기술 발전의 중심이 되는 IETF(Internet Engineering Task Force)를 주축으로 기존의 인터넷에 다양한 멀티미디어 트래픽을 보다 효율적으로 수용하기 위한 연구가 진행되고 있다.

본 논문은 Linux 기반 환경에서의 정책을 적용할 수 있는 COPS의 구현과 정책의 서버에 해당하는 PDP와 클라이언트인 PEP 구현, 서버와 사용자간의 인터페이스 구현에 초점을 두었다.

구현 방안으로 COPS 프로토콜을 이용할 수 있는 PDP 서버와 PEP 클라이언트 구현, PDP와 COPS 프로토콜과의 동작, PDP와 LDAP간의 인터페이스 구현 등 LDAP, COPS 프로토콜을 사용하여 정책 기반 시스템을 구축하고, 직접 QoS 서비스 제공을 위한 메커니즘을 연구 개발 목적으로 구현하였다.

향후 과제는 LDAP 서버와 관리자에 의해 정책 관련 데이터를 용이하게 사용할 수 있도록 어플리케이션 인터페이스와 다양한 서비스 체계에 동적으로 적응되도록 구현하는 것이 필요하다.

5. 참고문헌

[1] Boyle, J., Cohen, R., Durham, D., Herzog, S., Raja, R. and A. Sastry, "The COPS (Common Open Policy Service)Protocol", RFC 2748 <http://community.roxen.com/developers/ idocs/rfc/ rfc2748.html>, January 2000.

[2] Yeong, W., Howes, T., and S. Kille, "Lightweight Directory Access Protocol", March 1995.

[3] Braden, R., Zhang, L., Berson, S., Herzog, S. and S. Jamin, Resource ReSerVation Protocol (RSVP) Version 1 - Functional specification", RFC2205<http://community.roxen.com/developers/idocs/rfc/rfc2205.html>, September 1997.

[4] IntServ Working Group and [RFC 2211, 2212], http://www.ietf.org/html.charters/intserv-charter.html

[5] Yavatkar, R., Pendarakis, D. and R. Guerin, "A Framework policy Based Admission Control", RFC 2753, January 2000.